CLAIM AMENDMENTS

This listing of claims will replace all prior versions, and listings, of claims in the application:

1. (Currently Amended) A computerized method of managing a file system for a file server, comprising:

receiving a file operation that signals a reservation operation for reserving an additional number of blocks for storing a file of the file system, the file having a file size;

computing a first number of blocks needed to accommodate the file size;

subtracting from the first number of blocks a second number of blocks already allocated for the file and a third number of delayed allocated blocks for the file to obtain a fourth number of unallocated blocks needed to accommodate the file size; and

using the fourth number of blocks to perform a reservation of unallocated blocks for the file for later allocation.

2. (Previously Presented) A method as in claim 1, wherein the file system uses a write anywhere file system layout characterized in that data to be written are written to new blocks instead of being written to blocks previously allocated for said data.

3. (Original) A method as in claim 1, wherein the file operation that signals the reservation operation is a zero length write request.

4. (Original) A method as in claim 1, wherein the file operation that signals the reservation operation includes a parameter that specifies the file size.

5. (Previously presented) A method as in claim 1, wherein computing comprises:

determining a total number of direct and indirect blocks needed to accommodate the file size.

6. (Previously Presented) A method as in claim 1, further comprising:

setting a flag in an inode for the file that indicates blocks have been reserved for the file.

7. (Previously Presented) A method according to claim 1, wherein said using the fourth number of blocks to perform a reservation of unallocated blocks for the file for later allocation comprises:

checking that a number of available blocks in the file system is greater than the fourth number of blocks, wherein an error is returned in a case that the number of available blocks is less than the fourth number of blocks.

8. (Original) A method as in claim 7, wherein the number of available blocks in the file system is determined by subtracting a number of allocated blocks, a number of cached unallocated blocks, and a number of reserved blocks from a total number of blocks in the file system, and adding a number of reserved cached unallocated blocks.

9. (Previously Presented) A method according to claim 1, wherein said using the fourth number of blocks to perform a reservation of unallocated blocks for the file for later allocation comprises:

checking that a fifth number of blocks does not exceed a remainder of a quota for an owner of the file, wherein an error is returned in a case that the fifth number of blocks exceeds the remainder of the quota, wherein the fifth number of blocks comprises a difference between the first number of blocks and the second number of blocks.

10. (Previously presented) A method as in claim 1, further comprising releasing reservation of blocks as blocks are written to storage.

11. (Previously presented) A method as in claim 10, wherein releasing reservation of blocks further comprises decrementing the number of reserved unallocated blocks by a number of released blocks.

12-21. (Canceled)

22. (Previously Presented) A method according to claim 1, further comprising:

caching one or more blocks of the file in a buffer;

writing the one or more blocks to storage; and

decrementing the number of unallocated blocks by the number of blocks written to the storage.

23. (Previously Presented) A method according to claim 22, further comprising setting a caching flag for each block cached in the buffer.

24. (Previously Presented) A file server comprising a memory storing a computer program, a processor capable of executing the program, and a storage device capable of storing files of a file system under control of the processor, wherein the program comprises:

instructions to cause the processor to receive a file operation that signals a reservation operation for a file of the file system, the file having a file size;

instructions to cause the processor to compute a first number of blocks needed to accommodate the file size;

instructions to cause the processor to subtract from the first number of blocks a second number of blocks already allocated for the file and a third number of delayed allocated blocks for the file to obtain a fourth number of unallocated blocks to be reserved to accommodate the file size; and

instructions to cause the processor to use the fourth number of blocks to perform a reservation of unallocated blocks for the file for later allocation.

25. (Previously Presented) A file server according to claim 24, wherein the file system uses a write anywhere file system layout characterized in that data to be written are written to new blocks instead of being written to blocks previously allocated for said data.

26. (Previously Presented) A file server according to claim 24, wherein the file operation that signals the reservation operation is a zero length write request.

27. (Previously Presented) A file server according to claim 24, wherein the file operation that signals the reservation operation includes a parameter that specifies the file size.

28. (Previously Presented) A file server according to claim 24, wherein the instructions to cause the processor to compute comprise instructions to cause the processor to determine a total number of direct and indirect blocks needed to accommodate the file size.

29. (Previously Presented) A file server according to claim 24, wherein the program further comprises instructions to cause the processor to set a flag in an inode for the file, the flag indicating that blocks have been reserved for the file.

30. (Previously Presented) A file server according to claim 24, wherein the instructions to cause the processor to use the fourth number of blocks to perform a reservation of unallocated blocks for the file for later allocation program comprise instructions to cause the processor to check whether a number of available blocks in the file system is greater than the fourth number of blocks, and return an error in a case that the number of available blocks is less than the fourth number of blocks.

31. (Previously Presented) A file server according to claim 30, wherein the processor determines the number of available blocks in the file system by subtracting a number of allocated blocks, a number of cached unallocated blocks, and a number of reserved blocks from a total number of blocks in the file system, and adding a number of reserved cached unallocated blocks.

32. (Previously Presented) A file server according to claim 24, wherein the program further comprises instructions to cause the processor to check whether a fifth number of blocks does not exceed a remainder of a quota for an owner of the file, and return an error if the fifth number of blocks exceeds the remainder of the quota, wherein the fifth number of blocks comprises a difference between the first number of blocks and the second number of blocks.

33. (Previously Presented) A file server according to claim 24, wherein the program further comprises instructions to cause the processor to release reservation of blocks as blocks are written to storage.

34. (Previously Presented) A file server according to claim 33, wherein the instructions to cause the processor to release comprise instructions to cause the processor to decrement the number of the reserved unallocated blocks by a number of released blocks.-

35-45. (Canceled)

46. (Currently Amended) A method comprising:

receiving at a storage server a request for a space reservation for a data set managed by the storage server; and

in response to the request,

computing a number of blocks needed to be reserved for the data set, and

reserving for later allocation a number of unallocated blocks equal to the computed number of blocks, such that a subsequent write operation associated with the reservation can complete the write request without prevention of completion of the write operation due to insufficient memory.

47. (Previously Presented) A method as recited in claim 46, further comprising:

performing a write operation to write data to the data set by

determining whether a space reservation has been performed for the data set, and

in response to determining that a space reservation has been performed for the data set, allocating one or more blocks for said data without determining whether enough blocks are available for completing the write operation.

48. (Previously Presented) A method as recited in claim 47, further comprising:

in response to determining that a space reservation has not been performed for the data set, determining whether enough blocks are available for completing the write operation prior to allocating any blocks for said data.

49. (Previously Presented) A method as recited in claim 46, wherein the storage server employs a methodology in which data to be written are written to new blocks instead of being written to blocks previously allocated for said data.

50. (Previously Presented) A method as recited in claim 46, wherein the request comprises a zero length write request.

51. (Previously Presented) A method as recited in claim 46, wherein said computing the number of blocks comprises:

computing a first number of blocks representing a number of blocks needed to accommodate a size of the data set;

computing a second number of blocks representing a number of blocks already allocated for the data set;

computing a third number of blocks representing a number of delayed allocated blocks for the data set; and

subtracting the second number of blocks and the third number of blocks from the first number of blocks to produce a fourth number of blocks representing the number of blocks needed to be reserved for the data set.

52. (Previously Presented) A method as recited in claim 46, wherein said reserving for later allocation the number of unallocated blocks comprises:

setting a flag in a first metadata container associated with the data set, that indicates blocks have been reserved for the data set.

53. (Currently Amended) A method as recited in claim 52, further comprising:

examining the flag during a subsequent write operation to determine whether blocks have [[bee]] been reserved for the data set.

54. (Previously Presented) A method as recited in claim 52, wherein said reserving for later allocation the number of unallocated blocks comprises:

incrementing a reserved block count in a second metadata container associated with the data set by the number of blocks needed, the reserved block count indicating how many unallocated blocks have been reserved for data sets managed by the storage server.

55. (Currently Amended) A storage server comprising:

a processor;

a network interface through which to communicate with a remote client;

a file system; and

a storage device storing code which, when executed by the processor, causes the storage server to execute a process that includes

receiving a signal corresponding to a request for a space reservation operation for a file in the file system;

computing a first number of blocks needed to be reserved to accommodate the file;

reserving for later allocation a fourth number of unallocated blocks in the file system such that the fourth number is calculated by subtracting from the first number of blocks a second number of blocks already allocated for the file and a third number of delayed allocated blocks for the file; and

~~equal to the number of blocks needed to be reserved to accommodate the file.~~

performing a write operation to write data to the file by

determining whether a block reservation has been performed for the file, and

in response to determining that a block reservation has been performed for the file, allocating one or more blocks for said data in the file system without determining whether enough blocks are available in the file system for completing the write operation.

56. (Previously Presented) A storage server as recited in claim 55, wherein said process further comprises:

in response to determining that a block reservation has not been performed for the file, determining whether enough blocks are available in the file system for completing the write operation prior to allocating any blocks for said data in the file system.

57. (Previously Presented) A storage server as recited in claim 55, wherein the file system employs a methodology in which data to be written are written to new blocks instead of being written to blocks previously allocated for said data.

58. (Previously Presented) A storage server as recited in claim 55, wherein the signal represents a zero length write request.

59. (Previously Presented) A storage server as recited in claim 55, wherein said computing the number of blocks needed to be reserved to accommodate the file comprises:

computing a first number of blocks representing a number of blocks needed to accommodate a size of the file;

computing a second number of blocks representing a number of blocks already allocated for the file;

computing a third number of blocks representing a number of delayed allocated blocks for the file; and

subtracting the second number of blocks and the third number of blocks from the first number of blocks to produce a fourth number of blocks representing the number of blocks needed to be reserved to accommodate the file.

60. (Previously Presented) A storage server as recited in claim 55, wherein said reserving for later allocation the number of unallocated blocks in the file system comprises:

setting a flag in a first metadata container associated with the file, that indicates blocks have been reserved for the file.

61. (Currently Amended) A storage server as recited in claim 60, wherein said process further comprises:

examining the flag during a subsequent write operation to determine whether blocks have [[bee]] <u>been</u> reserved for the file.

62. (Previously Presented) A storage server as recited in claim 60, wherein said reserving for later allocation the number of unallocated blocks in the file system comprises:

incrementing a reserved block count in a second metadata container associated with the file by the number of blocks needed, the reserved block count indicating how many unallocated blocks have been reserved for files in the file system.

63. (New) A computerized method of managing a file, comprising:

receiving a write request for a file;

determining a desired number of blocks in which to store the file;

determining a number of previously reserved blocks by adding allocated and nonallocated blocks previously associated with the file;

determining an additional number of blocks by subtracting the number of previously reserved blocks from the desired number of blocks in which to store the file;

reserving the additional number of blocks; and

writing the file after successfully reserving the additional number of blocks.

64. (New) The method of claim 64, wherein the desired number of blocks is the minimum number of blocks required for storing the file;

65. (New) The method of claim 63, wherein the file is cache storage.

66 (New) The method of claim 63, wherein the reserving the additional number of blocks includes setting a flag in an inode for the file.